Software Tools COMMUNICATIONS

NUMBER 8 MAY 1982

JULY MEETING

1. When and Where?

Friday, July 9, 1982
Copley Plaza Hotel
Boston, MA

The Software Tools Users Group meeting will again be held in conjunction with the Usenix (and /usr/group) meeting, July 6-9, 1982. (Note that the Tools group will meet on the last day of the meeting, not the first, as before.)

2. Registration Fees:

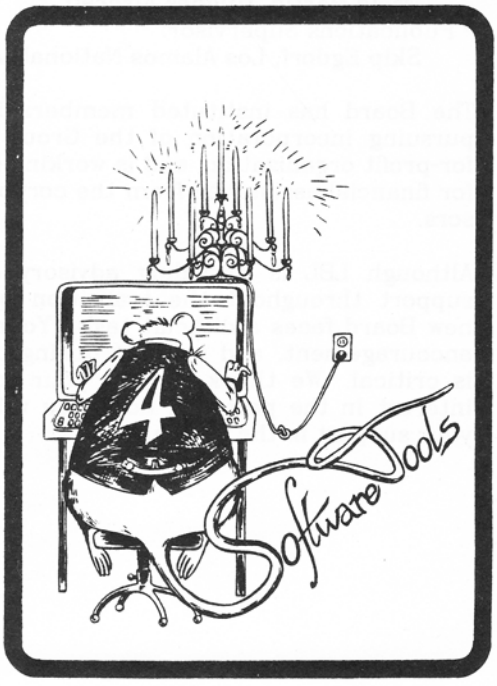
There will be one overall registration fee for the combined Software Tools, Usenix, and /usr/group sessions. If you have specific registration questions, contact:

Suzanne MacNary
617-497-2964

3. Agenda:

Session topics include, but are not limited to:

* The Virtual Operating System Approach
  Description of the design and implementation of projects using the virtual operating system approach; research topics generated by the approach
* New and Enhanced Utilities
  New utilities designed to enhance the program development environment; utilities which extend the environment (graphics, data management, etc.)
* Technical and Portability Issues
  Specification of new primitives; difficulties in implementations which affect the overall package; standards; networking
* Commercial Concerns
  Technical presentations describing commercially available implementations; applicability of package to commercialization; user attitudes toward commercialization
* Future Directions
  Suggestions for further development; research directions; user group organization and concerns



4. Proceedings

The conference committee intends to produce a proceedings consisting of copies of short (less than 10 pages) papers by the authors on the subject of their presentation, as well as all abstracts. Submission of a paper is not required, although it is strongly recommended. Papers will be collected in camera-ready form at the conference.

5. Speakers

There are still a few spots open on the agenda. Anyone interested in speaking should immediately contact:

Neil Groundwater
703-893-6140

STATUS OF THE USERS GROUP

The Software Tools Users Group was formed by Lawrence Berkeley Laboratory in 1978 as an experiment in user-defined software standards, including the collection, evaluation, and distribution of software variants. The goal was to create a uniform operating system interface for the full spectrum of vendor systems. If the experiment succeeded, the group was expected to become self-sustaining.

During the past years the group has been successful -- holding well-attended meetings, publishing newsletters and software catalogues, establishing standards, and distributing tapes. There are now almost 2000 members. However, the activities of managing and supporting such a large effort could no longer be considered research. Late last year, LBL decided to urge the fledgling out of the nest by cutting back support from the research program and encouraging support from individuals and private industry.

The decision was announced in the November 1981 newsletter and discussed at the Santa Monica meeting. At that time, support was offered by DEC, Cray, Unicorn Systems, and (later) Amdahl. A Board of Directors was elected:

* Co-ordinator:
  David Stoffel, Scion Corp.
* Liaison:
  Neil Groundwater, Analytic Disciplines
* Software Co-ordinator:
  David Martin, Hughes Aircraft
* Publications Supervisor:
  Skip Egdorf, Los Alamos National Laboratory

The Board has instituted membership fees, is pursuing incorporation of the Group as a not-for-profit organization, and is working out details for financial assistance from the corporate sponsors.

Although LBL is providing advisory and other support throughout the transition period, the new Board faces a difficult task. Your patience, encouragement, and support during this period is critical. We thank you for your tremendous interest in the past and hope you will continue your support in the future.

MEMBERSHIP IN STUG

Membership in the Software Tools Users Group now requires a registration fee. Payment of the yearly fee will entitle you to 4 newsletters per year, announcements of meetings, and rights to purchase the distribution tapes and make use of other services offered by the organization. If you feel the Tools movement is worth your support, send back your registration forms promptly.

REQUEST FOR BIDS

The new Board of Directors is soliciting proposals from organizations or individuals wishing to contract for any of the following services:

* Clerical Support
* Newsletter Production
* Software Collection and Preparation
* Tape Distribution
* Software Catalog Production

Anyone interested should contact:

David Stoffel
Scion Corp.
12310 Pinecrest Road
Reston, VA 22091
703-476-6100

CALL FOR TAPE SUBMISSIONS

The primary goal of the software tools virtual operating system movement is to make available a uniform user interface across system boundaries. The Software Tools Users Group serves as the focal point for collecting and screening enhancements and extensions, recommending those to be included in the standard interface. The first "basic" tape provided the core, or minimum functionality required. Now, with over 50 systems supporting the package, we are ready to collect and evaluate new extensions and variations, eventually releasing a second tape.

The next tape will again have two parts: A) those new utilities and library extensions considered basic, or minimum, for the (now extended) virtual operating system interface; b) variations to existing utilities and not-yet-standardized extensions, being distributed for evaluation and experimentation. As specific extensions become accepted, they will be moved up to the first part of future tapes.

The Users Group Role and Your Responsibility

The Software Tools Users Group is experimenting with some very unique issues: can users develop and maintain their own software without reliance on a vendor or supplier; and, can the resulting proliferation of variants be managed and controlled?

The first tape was a major step in successfully dealing with these issues. However, that tape took 8 months of effort to compile and prepare. The next tape will have the advantage that submissions will (hopefully) be built upon a common base. However, we no longer have the resources necessary to edit a tape as we did before. With your help, we can minimize the editorial work necessary to put together this next tape. Please, take the time to evaluate and clean up your own code, attempting to avoid the problems listed below. Remember, we the users are the support group--your valuable work may be lost to the community if you cannot donate a reasonably polished product.

Suggestions for Contributions

1) New utilities and extensions to the VOS are most desirable. Debugged and extended versions of the original package will also be accepted. However, because the research community has yet to develop the procedures and mechanical aids needed to merge sources developed at different sites, it is unlikely we will be able to hand-merge all variants into one acceptable version.

2) Extensions to the primitive set are greatly encouraged. Submit a "man" page and any other relevant documentation. When specifying new primitives, take the time to think them through carefully. Aim for implementability, generality, usefulness, and consistency both with existing primitives and with UNIX. It is admissible to suggest new capabilities for existing primitives, providing the need is ***clearly*** valid. The Primitives SIG will review all recommendations and make final decisions.

3) New (significant) additions to the library are also encouraged. Suggestions include environment handling, portable real-number handling facilities, argument handling routines, etc. (Please, no more variations on "scopy".)

Criteria for Acceptance

1) Must be written in Ratfor, Pascal, or C and be based on the standard software tools library and primitives (or their appropriate extensions). (C and Pascal versions may be kept on separate sections of the tape.) Material subject to copyright, license, or other control should not be submitted.

2) Portability
The submissions should be implementable on virtually any operating system. Machine-specific code or parameters should be implemented via macros or (possibly new) primitives. Non-standard language features should be avoided.

3) Consistency with style of existing utilities
Excessive featurism should be avoided. The syntax of flags and command should be consistent with previous usage, where applicable. Calling parameters of existing routines should not be changed. Argument handling should be consistent with prior usage (the utility should read from a list of file names as well as STDIN; utility should respond to a bare "?" with a usage message), etc.

4) Consistency with UNIX usage
Should be maintained unless there are convincing reasons why a change is appropriate. [This one is open to debate...]

Problems to Avoid

What follows is a list of the most severe problems we ran into when preparing the first tape:

1) Improper Data Typing
Variables and functions must be correctly typed and there should be no mixing of data types in the same arrays. This is not only bad programming practice, but has caused us considerable grief when attempting to port code to machines which store characters and integers differently. Correcting improper usage of data types accounted for much of the effort spent in putting together the first tape.

2) Use of machine-specific features or knowledge
Actual machine-specific system calls, or whatever, were generally avoided in the first submissions. However, the more subtle temptations such as using quoted strings and ascii character arrays interchangeably (because "they're stored the same on my machine"), or applying arithmetic procedures to seek addresses ("on my machine they represent byte counts"), or the

mixing of character and integer data types mentioned above ("they are the same on my system") forced us to rewrite, or even discard, many of the submissions to the first tape. It is extremely tempting to make use of efficiencies offered by the local system. However, the price one pays is portability.

3) Missing Library Routines
When writing utilities, one often comes up with new, generally useful routines which are subsequently added to the local library. This is fine. However, don't forget to include copies of these routines with your submission.

4) Improper Use of Existing Routines
Several people submitted utilities which required additional capabilities (and subsequent changes in calling parameters) for existing primitives or library routines. This is admissible providing the need is clearly valid and appropriate documentation accompanies. However, in most cases the changes could more correctly be done by building another routine on top of the existing primitive. Any future submittees requiring functional or syntactical changes to existing routines should be well-prepared to defend their suggestions.

5) Use of Non-standard Language Features
We were forced to edit many submissions which would not run through the standard ratfor compiler, or contained non-standard Fortran features.

Guidelines for Submissions

One archive file for each new utility/package of routines is easiest for us to deal with. The archive should contain:

* Additions to the standard symbols file required by the utility
* Additional library routines required, including their "man" pages
* "Man" pages for any additional primitives needed
* "Man" page for the utility itself, plus any tutorials or other relevant documentation
* Symbol definitions and common blocks needed for the utility
* Source code for the utility

Tapes can be either 800 or 1600 BPI; 512 characters per record (lines terminate by NEWLINE) is desirable but card images and other blockings are acceptable. Send tapes to:

Software Tools Users Group
Attn: Tape Submission
242-1259 El Camino Real
Menlo Park, CA 94025

Any organization making a (reasonable) contribution to the tape will receive a free copy of the completed tape. In addition, anyone donating code which takes us one day or less to implement will receive a Rat4 T-shirt, free of charge. Finally, anyone donating code which takes us more than a few days to implement will have his tape shipped to the outer reaches of Mongolia, at the contributor's expense.

SOFTWARE TOOLS ON UNIX - 4.1 BSD

There is now a version of the Software Tools Users Group basic tape implemented under UNIX - Berkeley 4.1 distribution. The primitives have been implemented as calls to the appropriate C library and system call routines. Everything specified by the users group has been implemented except the shell, which seemed superfluous. The local UNIX shell is used to perform I/O redirections.

Random writing on 4.1 BSD does not work correctly when interspersed with random reading because of system buffering problems. This is a known bug and there is evidently a fix for it 'somewhere'. Because of this, the editor as distributed by the Users Group would not work. We have provided an alternative version which uses an in-core buffer to maintain the lines (quite acceptable on a virtual memory machine) and is somewhat more enhanced than the regular tools editor.

Also included on the tape, though not yet included in the standard Software Tools package, is the Text Control System (TCS) developed by Neil Groundwater at ADI. TCS provides capabilities similar to SCCS, allowing a user to keep track of the evolution of source code.

The tape is being made available through the Software Tools Users Group. Executable images have not been included on the tape, to avoid licensing problems. None of the source code is licensed material.

NEW TAPE DISTRIBUTION PROCEDURES

The Users Group is in the process of expanding its tape distribution services. Besides having available the "basic" portable tape, it will also distribute specific machine implementations, providing they include all the utilities and versions of the primitives specified on the basic tape. Current offerings include:

VAX/VMS (1600 BPI, Files-11 format)
PDP 11 RSX-11M/IAS (800 BPI, FLX format)
VAX/UNIX 4.1 BSD (800 BPI, tar format)
UNIVAC 1100 (copy,gm format)

plus

Basic tape, blocked 2048 characters/record (800 BPI, ASCII)
Basic tape, blocked 40 card images/record (800 BPI, ASCII)

Anyone wishing to have the Users Group distribute her implementation should contact the address below.

Cost of tapes is now $50, plus $10 additional for special formatting and/or foreign mailing. Send checks to:

Software Tools Users Group
Attn: Tape Distribution
242-1259 El Camino Real
Menlo Park, CA 94025

SOFTWARE TOOLS ON HP1000

The full set of software tools primitives has been implemented for the HP1000 running RTE 4A, with versions for FTE 4B and RTE 6 to be completed soon. The utilities implemented include ratfor, format, the shell, ar, cat, crt, detab, echo, includ, mv, pr, rm, tail, tee, and wc, plus a much enhanced version of ed, written by Ellis Cohen. A distribution tape is available and will include:

* All running tools, with primitives for RTE 4A, 4B, and 6
* Source for all tools (except the editor), plus the primitives
* Documentation for the above
* The STUG Basic Tape
* Installation procedures

This is not an HP product and there probably will be no license charge for the tape. For further information, contact:

Larry Dwyer
Hewlett Packard, Bldg. 43D
11000 Wolfe Road
Cupertino, CA 95014

SOFTWARE TOOLS ON HP3000

The STUG set of software tools has been implemented for the HP3000. All primitives are completed, with the exception of READWRITE file access (soon to be finished). Most utilities are already running, with the remainder soon to be completed. A much enhanced version of ed, written by Ellis Cohen, replaces the standard editor. A distribution tape is available and will include:

* All running tools and primitives
* Source for all tools (except the editor), plus the primitives
* Documentation for the above
* The STUG Basic Tape
* Installation procedures
* Programmer notes on the primitives

There will be a (as yet undetermined) license charge for the primitives. For further information, contact:

Ken Poulton
3182 Greer Road
Palo Alto, CA 94303
415-857-1742 (after 6:30 PM)

A Bug in the Pascal Version of the Tools

Skip Egdorf
Los Alamos National Laboratory

There is a bug in the system interface package to the Berkeley UNIX (tm) Pascal system. When attempting to open a file for reading with the interface routine "open", the file is truncated. The interface in the Pascal Tools book works correctly with the Pascal interpreter (pi). However, when used with the Pascal compiler (pc) the bug surfaces.

The interface routine "open" uses the Pascal builtin "reset" to open the desired file for reading. Within the Pascal library, the procedure which does "reset" calls the UNIX stdio routine "fopen" which calls "open" (the UNIX system call). The link editor already knows about the routine called "open" in the Pascal program, and links back to it recursively. On this second call to "open", the arguments are of different types (intended for the UNIX system call). These new arguments cause the open call to try to "create" the file. This works since the "else" branch covers all modes but "IOREAD". The result is that opening a file for reading truncates the file!

The fix for this bug is to use a wrapper program which defines "open" as something else (I use "POPEN"), and translates all "opens" before the program is compiled. This bug will also exist

with "close", and is fixed in the same way.

THE SOFTWARE TOOLS AT NCAR

Ben Domenico
National Center for Atmospheric Research

Abstract

The Libraries and Software Group of the Scientific Computing Division (SCD) at the National Center for Atmospheric Research (NCAR) is implementing the Software Tools package on a Cray-1 running the COS operating system, two PDP 11/70s running Interactive Systems UNIX (tm), and an IBM 4341 running VM/SP (the latest version of VM/CMS). The majority of our effort has been focused on the IBM system where we have now implemented all the primitives except SPAWN and about a third of the tools. It is our understanding that the primitives for UNIX and COS are already written and available; we intend to install them as soon as we can get copies.

In this paper, some of the CMS implementation problems are discussed and a list of the bugs that were uncovered has been included. By the next STUG meeting, the VM/CMS primitives should be available for distribution.

The NCAR Computing Environment

NCAR has a wide range of computers and operating systems available for use by the staff and by the atmospheric science community around the country. Within the Scientific Computing Division (SCD), the major computational facilities are a Cray-1 running the COS operating system, a Control Data 7600 with a home-grown operating system, and an Ampex Terabit Memory mass storage system. These are linked to each other and to a set of distributed "front end" computers via a local network utilizing NSC Hyperchannel hardware and locally-developed software. Among the front end machines are two PDP 11/70s with Interactive Systems UNIX, another 11/70 with PWD UNIX, a VAX 11/780 with VMS, and an IBM 4341 running VM/SP. The 4341 and the two Interactive Systems machines are administered within the SCD while the other front end machines are the responsibility of other NCAR divisions.

Goals

Within the SCD we intend to bring up as much of the Tools package as possible on the Cray-1, the 11/70s and the IBM 4341. We are also encouraging the people in charge of the other systems on the network to do likewise. Since most of our recent effort has been concentrated on getting the Tools running on the IBM system, this paper is mainly a description of that effort.

Status of the Tools in IBM VM/SP

With the exception of SPAWN, the basic system-dependent primitives have all been implemented. There is a problem getting lower case arguments longer than 8 characters into a program, but this has been circumvented in a manner that is acceptable for the present -- at least until we can get the shell running.

The file manipulation and disk I/O primitives make use of a package of assembler routines, written originally by B. Lynn Irwin for the local network and later augmented by Phil Rasch for use in the Tools primitives. This package of Fortran-callable routines invokes the CMS macros for manipulating File System Control Blocks (FSCB's). Terminal I/O is still done using Fortran formatted I/O and characters are still stored internally in their ASCII representation, one character per integer word. However, the inmap and outmap functions have been replaced by two arrays for the ASCII/EBCDIC mapping.

We have the following tools running: echo, cat, include, ratfor, ed, roff (format), ar, rm, find, sedit, date, diff, wc, tail, fb, and field. The major work left is to get the SPAWN primitive going and implement the rest of the tools.

Implementing the Primitives

Many of the problems with the implementation of the primitives stem from the difference in philosophy between IBM and the "Tools" world. The most striking example of this difference is the fact that IBM assumes that anyone who keys in a lower case character has either made some horrible mistake or was forced to strike that key at gunpoint. This assumption rears its ugly head throughout the tools implementation process on CMS.

Lower case Arguments

As mentioned above, the implementation of MAKARG was complicated by the fact that CMS provides a facility for accessing parameters from within an executing program, but only after each argument has been converted to upper case and truncated to 8 characters. To get around this, we have implemented a version of MAKARG that reads the command line from a temporary file. This approach works in conjunction with an interactive "EXEC" (a very rough analog of a UNIX shell command file) which provides what we call the "tools environment". This EXEC takes each command line from the terminal and writes it to the temporary file before executing the command. In this way each tool can read the

unabridged, mixed case commands from the file during execution.

For experienced IBM users, this approach can take some getting used to, since they are not accustomed to such things as files with lower case names. Moreover, some of the CMS tools are not able to deal with lower case file names in a consistent manner. In the tools environment, the CMS editor XEDIT can be invoked to edit a file with a lower case name. However, when the SAVE function is executed to write the edited file to disk, the file is saved under the upper case name. Problems such as this are going to require more than "getting used to." We are hopeful that a more consistent interface will result as more of the Software Tools are made available and fewer of the IBM tools are used.

File Naming Conventions

Depending on how you want to look at it, CMS has either no file naming conventions or too many conventions. Certainly there is no consistent convention. Some of the CMS programs require the user to specify a file by giving three separate arguments -- the file name, file type, and file mode. Others require only the file name and type; the program then either provides a default file mode or looks for a file of that name and type among all the possible modes. A few programs allow the use of "wild card" characters in the file name; others do not. Finally, some programs (among them the Fortran compiler, assembler, and loader) consider it an error if the user specifies anything more than the file name; these programs insist on a specific file type and have their own algorithm for determining the file mode.

For the Software Tools programs, we have adopted the convention that files are specified as one argument with the file name, file type, and file mode separated by periods. (Period is not a legal character in a CMS file name, file type, or file mode.) At present the default file type is "TOOLS" and the default file mode is "A". Thus, "junk.FORTRAN" specifies a file of name "junk", type "FORTRAN", and mode "A", whereas "BUNK..C" specifies a file of name "BUNK", type "TOOLS" and mode "C". This convention turns out to be convenient in unarchiving files from the Software Tools tape. For example, the files rm.r and rm.doc are archived in file rm.TOOLS. The command "ar -x rm" then produces two CMS files: one has the name "rm", type "r", the other file has the same name but is of type "doc". Both are mode "A" by default.

Bugs

In the process of implementing the primitives, library routines and tools, we uncovered several bugs which we will simply list here, since we are not aware of any formal bug reporting mechanism. These reports pertain to the version of the tape obtained at the Software Tools Users Group meeting held in San Francisco in January of 1981. [Editor's note: this is the same version of the Basic Tape that is still being distributed.]

edin
: The dummy argument "cursav" in "doglob" was changed to avoid a conflict with a variable of the same name in the included file "clines".

match
: The dimension of "junk" was changed from 9 to 10 to correspond with the usage in "amatch".

diff
: In "gen_script", a "call putint" statement was missing its second argument. An argument of 0 (zero) was inserted.

ar
: In the "gethdr" routine a FOLDF macro was inserted at the beginning of the line containing the "call foldf", so that setting the FOLDF macro OFF worked properly. The code for opening the archive file had to be changed in the case where the file did not previously exist, because CMS does not acknowledge the existence of a file (even a file which has been opened) until a record has been written into the file. This experience raised some questions about the proper behavior of the "open" and "create" primitives when an attempt is made to access a non-existent file with mode READ. In other words, we could use a more definitive statement of the circumstances under which these primitives should return the ERR error code.

fb
: The data statements in "fbargs" which set the value of variables in named common were moved to a block data subprogram.

note
: The order of the dummy parameters in the temporary version was reversed to match the specs. Since we built the primitives by modifying the temporary versions, this bug caused a lot of consternation.

ed
: In the routine "setbuf", the variable "scrfile" was changed to "scrfil" in the references to the "create" function.

macro
: The "puttok" routine was missing from the file macro.r. We took the version of puttok from ratfor, but the resultant macro processor seemed to behave strangely in that it didn't do the substitutions if the string was

followed by a period. Further investigation is necessary.

roff (format)

If the files specified in the argument list cannot be found, roff takes its input from the standard input. This can be confusing if one is not aware of what is happening. Also, the -s argument doesn't work when used on our Diablo printers.

ed

The "$n" buffers seem to work nicely if one is in need of "write only" storage space. More specifically, one can use them with the w (write) command, but not with any of the others. We have not fixed this yet but it appears that several more calls to look4 are needed.

generic bugs

In several of the temporary primitives, we had to remove implied do-loops from data statements.

CMS Software Tools Sources

At NCAR we intend to have our version of the primitives documented and ready for distribution by the next Software Tools Users Group meeting. For the edification of CMS buffs, we are using the FORTVS compiler, but we are running it at language level 66, so that it processes standard Fortran66 code.

At present, we are aware of the following sources of information, advice, and/or primitive level code for the Software Tools on various versions of the VM/CMS operating system:

Ben Domenico
NCAR
P. O. Box 3000
Boulder, CO 80307
303-494-5151 x559

William J. Donovan, Jr.
12815 S. W. 112 Terrace
Miami, Florida 33186
305-385-3764

Ed Happ
Interactive Data Corp
486 Totten Pond Road
Waltham, MA 02154
617-895-4225

Leo C. Nordhuisen
Philips ISA - TIS/CARD
Building SAQ 2524
5600 MD
Eindhoven, Netherlands
Netherlands 40-413941

Mike Norred
ERTEC Rocky Mountain, Inc.
1746 Cole Blvd.
Golden, CO 80401
303-277-0900

TOOLS ON DATA GENERAL

Jon Hanshew
President, CompuCode
6147 Aspinwall Road
Oakland, CA 94611
415-339-9463

Introduction

Many of the software tools have been ported to Data General computers by CompuCode. CompuCode's objective is to port the tools to a DG environment in such a way as to make porting them between the various DG operating systems and Fortran versions very easy. The tools which have been brought up are available from CompuCode.

The development work was done on a DG Nova 4/C (copyright Data General Corp.), 10 Mbyte hard disk, a CRT and a letter quality printer.

Data General offers several operating systems (MPOS, DOS, RDOS, AOS, and AOS/VS), the most common being RDOS and AOS; there are at least four Fortrans (MP/FORTRAN IV, FORTRAN IV, FORTRAN 5, and FORTRAN 77). The various versions of operating systems and languages are fairly compatible, but a program running under one operating system/Fortran combination will probably not run under another combination unless care has been taken to restrict language usage to the subset of available options common to all versions. The operating systems offer many good and useful utilities. In fact, all of the tools brought up were developed under them. However, as a UNIX (tm) user, their lack of certain capabilities left me frustrated. For example, the DG editor, SPEED, has no pattern matching capabilities and its command set is nonintuitive. Consequently, when I heard about the Software Tools, I obtained the Basic Tape and embarked on bringing up the tools on my own system.

The following is an account of what I learned while bringing up the tools on a small DG system. The sequence of topics follows, roughly, the sequence in Debbie Scherrer's "Cookbook" for bringing up the tools.

COPY

The first tool to bring up is COPY, a program that simply copies an input file to an output file. This is not a program that a DG user would actually ever use (RDOS utilities exist which are faster) but it is useful for testing I/O primitives. If your system uses some character set other than ASCII, then the external characters have to be mapped into the internal ASCII characters. Since DG uses ASCII, the mapping doesn't have to occur. However, the Tools's internal representation for character data is one right justified byte per machine word, while DG's external representation is a packed byte sequence. Regardless of how one might feel about this scheme, it is not something that is wise to change.

Ratfor Bootstrap

The next tool to bring up is the ratfor bootstrap. I had a number of problems, many DG specific ones and some program bugs. DG Fortran allows very long program names, but they must be distinct within five characters instead of the standard six. Fortunately, almost all software tool subroutines are distinct within five characters anyway. The ones that aren't must be altered to conform to the DG standard.

Another DG specific problem relates to DG's reserved words. For example, TYPE is a reserved word in DG Fortran. TYPE is also a subroutine name in the Software Tools. I ultimately dealt with this problem by having the Ratfor preprocessor translate all such problematic words into ones that the Fortran compiler found less offensive. "type" became TTYPE for example. Since the bootstrap cannot be run through the preprocessor, these changes must be edited out of the bootstrap by other means. I did global changes using the DG editor, SPEED. Similar problems arose with conflicts between DG library subroutine names and software tool subroutine names. Both systems want to use "OPEN". The same approach solved this problem well.

Another DG implementation problem relates to the requirement that all variables initialized via DATA statements must be declared COMMON. Since DG allocates non-COMMON variables dynamically, the only available static storage is common storage. Many Software Tools variables are DATA initialized, in particular, string variables. All DATA statements in the bootstrap must be provided with corresponding COMMON declarations. Later on when the bootstrap is up and running, it must be altered to automatically make the COMMON declarations for strings.

After successfully getting all bootstrap subroutines to compile, I learned that I had another problem as soon as the size of the entire program was reported. It was way too large for my available space. RDOS takes up 18kb. The DG Fortran library programs take up much more. The space required for the preprocessor was also great. It had to be squeezed. One way to reduce the bootstrap's size was to eliminate all mention of Fortran I/O. This reduced its size by about 2K, but that wasn't enough. It had to be overlayed. Deducing the bootstrap's structure to the point that it could be successfully overlayed required me to develop automated techniques for cross-referencing the large collection of programs that make up the ratfor bootstrap. It is ironic that I embarked on this project motivated by a desire to get a familiar editor running on a new machine, but to accomplish this I ended up learning how to use my existing editor in previously unimagined ways.

In addition to overlaying, all large arrays had to be disk buffered. The space reserved for macros had to be sent out to disk. DG provides subroutines which do direct block I/O that are very fast. To locate or insert a macro definition, the number of the disk block is calculated, the block fetched and the data are inserted or collected as required.

After overlaying, eliminating Fortran I/O, and disk buffering the large arrays, one mysterious problem remained. This one turned out to be a Software Tool program error. Evidently it doesn't matter on some machines whether programs are declared subroutines and called as functions or the reverse, but on DG systems it creates havoc. Similar problems occurred while bringing up other tools. I now routinely screen each new tool's source code for such occurrences before passing it through the preprocessor and compiling it.

Reading Command Arguments

The Cookbook for this file begins, "Now the real work begins...". If this was to be harder in comparison to the bootstrap, then I was in real trouble. But it wasn't. RDOS provides a disk file containing the command line arguments. It should be noted that the RDOS command line interpreter uses right and left arrows, therefore they are unavailable for use in Software Tools command lines.

The Includ Tool

After bringing up and testing the INCLUD tool, I learned that the ratfor bootstrap had the include capability. It just had to be implemented. I recommend the inclusion of the INCLUD facility into the ratfor bootstrap immediately after it has been separately tested.

The Text Formatter

I had to completely rewrite the "data set" related code for a couple of reasons. One was that it had some bugs, the other was that those large arrays would not fit into my little machine. Once again I had to disk buffer the large arrays and overlay to get it to fit. A CRT terminal interface had to be added as well.

The Editor

The main change to this program was to add a CRT interface and to speed up the I/O.

When running under the portable primitives, the editor runs very slowly. This is where I wrote my I/O assembly subroutines. A requirement of these primitives was DG operating system independence. Some DG operating systems use carriage returns for record terminators in serial text files while others (MPOS and AOS) use line feeds. By making the line terminators and their followers parameters to the I/O primitives, the need to reprogram the primitives for the various operating systems was avoided.

Spell

This program uses some gigantic arrays and some gigantic disk files. As usual, all large arrays had to be disk buffered, but by this time the procedure was fairly routine. Array references are identical to function calls when they appear on the right side of an assignment. I used an editor macro to change array value setting assignment statements to subroutine calls. DG Fortran also provides the capability of passing a function as a subroutine parameter when it has been declared "EXTERNAL" in the called subroutine. By systematically implementing these observations, the large arrays were successfully disk buffered. Since the arrays became non-volatile when they were buffered, it was a simple matter to speed up the program dramatically after it had been initialized once. Further performance improvements were achieved by noting the starting and ending ranges for the initial letters of words to be looked up and limiting the search for the word in the dictionary index to that range.

One program bug deserves mention. For some reason, the apostrophes in the spelling list are out of ASCII sequence. Words containing apostrophes must be re-edited into the appropriate places.

SEND IN

YOUR

MEMBERSHIP

FEES

NOW

Date: _____

SOFTWARE TOOLS USERS GROUP

APPLICATION FOR MEMBERSHIP

Name: _____

Address: _____

_____

_____

_____

Phone: _____

Network Address: _____

Machines and systems on which you have the Software Tools package running:

_____

_____

Utilities/library functions you have implemented.

_____ The standard package (as distributed by STUG)

_____ The original package (Kernighan-Plauger)

Additional/Other: _____

_____

_____

Other systems in which you are interested in obtaining/implementing Tools packages:

_____

Special interests: _____

_____

Check here if you do NOT want this information available to other members _____

Membership fees are $15 per year. Add $5 for overseas airmail, if desired. Make checks payable to the "Software Tools Users Group" and mail to:

Software Tools Users Group
242-1259 El Camino Real
Menlo Park, CA 94025